

PCTWORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : G06F 15/16	A2	(11) International Publication Number: WO 98/25210 (43) International Publication Date: 11 June 1998 (11.06.98)
(21) International Application Number: PCT/US97/22439 (22) International Filing Date: 3 December 1997 (03.12.97) (30) Priority Data: 08/762,186 4 December 1996 (04.12.96) US (71) Applicant: GIGANET, INC. [US/US]; Suite 108, 2352 Main Street, Concord, MA 01742 (US). (72) Inventors: FOLLETT, David, R.; 120 Cobleigh Road, Boxborough, MA 01719 (US). GUTIERREZ, Maria, C.; 19 Willow Street, Concord, MA 01742 (US). PROHASKA, Richard, F.; 14 Kings Row, North Reading, MA 01864 (US). (74) Agents: LOGINOV, William, A. et al.; Cesari and McKenna, LLP, 30 Rowes Wharf, Boston, MA 02110 (US).		(81) Designated States: AL, AU, BA, BB, BG, BR, CA, CN, CU, CZ, EE, GE, HU, IL, IS, JP, KP, KR, LC, LK, LR, LT, LV, MG, MK, MN, MX, NO, NZ, PL, RO, SG, SI, SK, SL, TR, TT, UA, UZ, VN, YU, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG). Published <i>Without international search report and to be republished upon receipt of that report.</i>
(54) Title: COMPUTER INTERFACE FOR DIRECT MAPPING OF APPLICATION DATA		
(57) Abstract		
<p>A computer interface system for communicating between computers along designated circuits is provided. An interface unit is provided in each host computer. Each unit includes a memory map that stores a map of physical addresses that correspond to virtual addresses for each application involved in a communication link. Each transfer of data is designated by a circuit that is established using the ATM protocol. The circuit is recognized by each unit involved in the communication and facilitates direct access of each host computer's main memory with minimum intervention from the operating system. A novel dynamic buffer management system is also provided.</p>		
BEST AVAILABLE COPY		

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakhstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LJ	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

COMPUTER INTERFACE FOR DIRECT MAPPING OF APPLICATION DATA

FIELD OF INVENTION

5 This invention relates to communications among applications running on different computers and computer subsystems. Specifically, it relates to the direct mapping of application data on one computer to physical memory addresses in another computer, and more particularly a switching arrangement that performs memory addressing functions between at least two networked computers substantially free of operating system intervention.
10

BACKGROUND OF THE INVENTION

 The transfer of data and instructions between data processing devices in a cluster or network is accomplished using interface units included in each device.
15 Such data processing devices can include, for example, computers and shared I/O devices such as disk servers and communication links to external networks. A link that typically transmits data among these units in serial form is used. Switches may be provided to route the data particularly when three or more devices are linked.

 At any time, a number of data transfers may be in progress among application
20 programs running on different computers. A number of networking arrangements have been implemented to accomplish these transfers. In general, such conventional networking approaches have failed to provide the desired combination of low latency, high speed and low cost.

 It is, therefore an object of this invention to provide a low-cost computer interface system that provides significant reductions in latency and increases in the speed
25 of data transmission between linked data processing devices. This interface system should use on-board resources efficiently to minimize hardware costs.

SUMMARY OF THE INVENTION

 This invention overcomes the disadvantages of the prior art by providing each
30 host data processing device (e.g. computer) with an interface unit that stores, for each

open communications circuit involving an application in the host device, a map identifying the virtual addresses specified by the application with corresponding physical addresses in the host's main memory. These physical addresses are the locations in the host device of data to be transferred between the application and a specific application running on another device. Data transfers occur along virtual circuits that each carry a unique identifier that facilitates data routing between computers. Each interface unit includes a series of registers or "memories" that are indexed or "mapped" in terms of the circuit identifiers.

The interface unit can, thus, transfer data to and from the main memory without resorting to time-consuming calls to the host operating system for protocol processing or to translate virtual memory locations provided by the application to physical memory addresses in the main memory. In addition, the registers in the interface unit can be used to retain protocol state information on a circuit-by-circuit basis. This facilitates protocol acceleration.

The invention also provides a novel buffer arrangement within the interface units and within switches used when three or more host data processing devices are involved. Buffer space is assigned dynamically to data transmission circuits passing through the interface, and switch, thus minimizing the required size of the buffer memory and lowering hardware costs. This technique also enables various "fair" and/or adaptive scheduling algorithms to be used.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects and advantages of the invention will become clear with reference to the following detailed description as illustrated by the drawings in which:

Fig. 1 is a block diagram of a network of computer systems that are linked in communication with each other according to this invention;

Fig. 2 is a more detailed block diagram of a network interface unit according to this invention;

Fig. 3 is a more detailed block diagram of another network interface unit according to this invention; and

Fig. 4 is a block diagram of a switch according to this invention.

5

DETAILED DESCRIPTION

Fig. 1 illustrates a basic clustered computer system according to this invention. By way of example, a cluster having three computers is illustrated. Although computers are shown in this example, one or more of the computers can be another data processing device such as a disk server or communication link to an external network.

10 The use of this system with such other devices is, therefore, expressly contemplated.

Reference shall be made generally to computer 30a for the purposes of this description. It shall be assumed that computer 30a is interconnected with computers 30b and computer 30c. As used herein, like reference numbers having a suffix "a", "b" or "c" shall refer to like components in each of the respective computer systems 30a, 30b and 30c. It is contemplated that these host computers 30a, 30b and 30c can differ in architecture so long as they are otherwise compatible to communicate with each other in a manner to be described below. A disk server 35 is also illustrated. Furthermore, as used herein, the term "data" when occurring in connection with the interface unit of this invention shall refer to both control (e.g. TCP/IP header information) and raw data. The interface of this invention includes framing functions that convert both control and data strings into ATM-framed data for transmission between data processing devices. The framed data is reconverted to control and data strings on the receiving end using the same functions.

Computer 30a comprises a CPU or central processing unit 40a linked by a parallel system bus 42a to a main memory 44a. Within the main memory 44a are blocks of locations that contain an operating system 46a and a series of computer application programs 48a.

An I/O sector 50a can include disks and other peripheral devices 52a and a network interface unit 54a according to this invention, all coupled to the system bus 42a. The interface unit 54a enables communication with the computers 30b and 30c

and the disk server 35. Computer 30b includes its own network interface unit 54b connected to its system bus 42b and computer 30c includes a network interface unit 54 connected to its system bus 42c. In a cluster in which only two computers (e.g., computer 30a and computer 30b) are connected, a direct interconnection 80 (shown in phantom) can be provided between the interface units 54a and 54b. Conversely, where multiple (e.g., three or more) computers are linked, a switch 82 is provided to route signals among the devices in the cluster. This disk server 35 is also connected in the cluster through the switch 82. It includes its own network interface unit 55 that is identical in architecture to the other units 54a, 54b and 54c. The direct connection 80 between two computers and the multiple links 84, 86, 88 and 89 among three or more computers (and the disk server 35) may take any suitable form such as LVDS (low voltage differential signaling) links or fiber optics links.

Figs. 2 and 3 show the network interface units 54a and 54b respectively, in greater detail. We shall now describe operation of the network interface units in greater detail, assuming first that the units 54a are interconnected by the serial link 80.

The link 80 is operatively connected to the network interface units 54a and 54b through respective transceiver/serial-parallel converters 90a and 90b. The transceivers 90a and 90b can both send and receive data for two-way communication over the link 80.

The preferred mode of data transfer between the network interface units is the well-known Asynchronous Transfer Mode (ATM). In ATM, messages are transferred in packets or "cells". All communications are framed in terms of the ATM UNI (User to Network Interface) standard. One or more virtual circuits are set up among application programs, running on different data processing devices (computers 30a, 30b and 30c and disk server 35 for example). Each data cell transmitted through the network has a header that includes a virtual circuit identifier (VCI) of the virtual circuit. In a multi-link circuit, there may be a different VCI for each link. Each circuit is established in full-duplex mode, with bi-directional data transfer between host devices occurring under the same circuit identifiers. The virtual circuits and their corresponding VCI's are generally established when the system boots up based upon the communication requirements of the applications present in each of the data processing

devices. Further virtual circuits may be established and existing circuits discontinued dynamically as needed by application programs in the cluster.

While ATM is described as the preferred communication standard according to this embodiment, it is expressly contemplated that other communication standards and protocols can be used in conjunction with the interface system according to this invention. For example, the TCP/IP protocol can be used. Appropriate framing/deframing modules that are well-known and commercially available can be employed to establish this communication standard in connection with the system of this invention. Thus, while the ATM standard is shown and described herein, TCP/IP, encryption standards and other compatible communication protocols can be substituted and the description is taken to include these alternative protocols.

When an application 60a running on computer 30a, for example, is to establish a circuit to a particular application 60b running on computer 30b, the application 60a informs the operating system 46a that the contents of a block (or blocks) of locations in memory 55a, identified by virtual memory addresses used by that application, are to be used in transmissions to and/or receipt from the application 60b, running on computer 30b. The application also communicates to the operating system the priority of the circuit, the protocol processing type, relevant flow control parameters and relevant transfer optimization parameters.

A switch controller 89, that is preferably implemented as a software application on the computer 30a receives instructions from the operating system to establish virtual circuits. It communicates with the network through the network interface unit 54c. The controller 89, having received a request for establishment of a circuit from the operating system 46a, opens communication with the target computer's operating system 46b. It also communicates with the switch 82 in a manner to be described below. The controller 89 instructs each of the operating systems 46a and 46b to utilize a specific VCI.

When circuits are established, corresponding pointers, each indexed by VCI, are established in a *transmit work queue* pointer register 100a of the interface unit 54a. Each pointer points to a transmit work queue 101a associated with a particular

-6-

application 60a within the host main memory 44a that is reserved for storage of transmit work queue information for the corresponding VCI. Each entry in the transmit work queue 101a stored in the main memory contains the start locations of control (e.g. TCP/IP) and data information and the amount of data to be transferred.

5 As transfers are accomplished, the pointer in the register 100a is advanced to the next entry in the register by the interface unit 54a.

The operating system 46a loads into a mapping memory 95a in the unit 54a a map of the virtual addresses provided by the application 60a to the corresponding physical memory addresses in memory 55a. The physical addresses being mapped are
10 normally locked in the host by the operating system so as not to be overwritten by other applications running on computer 30a. The memory 95a includes a separate map for each virtual circuit through the interface unit 54a. For each virtual circuit there is an associated transmit work queue 101a and a receive work queue 103a (described below) that is stored in the host main memory 44a. The mapping memory
15 95a, transmit work queue pointer register 100a and receive work queue pointer register 110a (also described below) are in the "address space" of the system bus 42a and can therefore be directly addressed by the operating system 46a and application 60a, respectively. Applications can, thus write directly to and read from these registers at any time.

20 At the receiving end of the transfer, the target application 60b has supplied the operating system 46b with the virtual addresses in memory 44b that are to receive and transmit data in the subject circuit. Operating system 46b loads a mapping memory 95b on the interface unit 54b with a map of these virtual addresses to the corresponding physical addresses in main memory 44b and locks the physical addresses in the
25 host operating system 46b to prevent inadvertent overwrite. For each virtual circuit there is an associated *receive work queue* 103b. Each entry in the work queue includes the beginning virtual addresses in the memory 44b to which control and data are to be transferred and the number of bytes of memory space following these addresses. An associated entry in a receive work queue pointer register 110b is provided
30 on the interface unit 54b. This entry contains a pointer to the next entry in the work queue 103b for that circuit. The pointer is initially set by the application 60a and then

advanced by the interface 54b to the next work queue entry after completion of the work specified in the entry. A notification technique is used to inform the application 60b of the arrival of control information and data. This technique can include a well-known polling and/or interrupt routines.

5 Since each virtual circuit handles data transmissions in both directions, e.g., from the application 60a to the application 60b and from the application 60b to the application 60a, the reverse setup procedure also takes place. The application 60a also makes the appropriate entries in its receive work queue pointer register 100a, receive work queue 103a and the transmit work queue pointer register 100b. The application
10 60b makes appropriate entries, when needed, in its transmit work queue 101b and associated pointer register 100b.

 The switch 82 is set up by the controller 89 concurrently with the interface units. The procedure is described below. Data transmitted over the links 80-86 is in ATM cells of 48 data bytes and 5 header bytes each. The instantaneous rates at which
15 data enters the interface unit 54a and at which it is transmitted from the unit will ordinarily differ. It is also desirable to transfer larger blocks of data in and out of the host memory and to transfer smaller cells over the network link. Moreover, a plurality of virtual circuits will ordinarily be open at any given time, requiring the unit to mux and demux the circuits in real time. Accordingly, a buffering arrangement is included in
20 the interface units. Specifically, we include a buffer memory 120a into which incoming data is loaded and from which outgoing data is retrieved. While any previously known buffer arrangement can be used, we prefer to use the dynamic buffer arrangement described below, since its size can be substantially less than that of other arrangements. First, however, we shall describe the manner in which the invention ac-
25 complishes a transfer of the data between applications running on different computers.

 Context memories 140a, b and c is provided in the interface units 54a, b and c. The context memory 140a, for example, contains an entry for each circuit or potential circuit occupying space in the buffer memory 120a. Each entry is indexed by its VCI and includes the priority level of the circuit, the number of buffer memory bytes oc-
30 cupied by the circuit, flow control parameters, host transfer optimization information, protocol state information, the receive and transmit head and tail addresses of the cir-

cuit's data in the thread memory 130a and the host notification mechanism (e.g. interrupt or polling routine).

Each network interface unit 54a, b or c also includes an active circuit memory 160a, b or c containing a list of all the active circuits with data in the dynamic buffer memory 120a, b or c. In other words any incoming data or outgoing data (appearing in the transmit work queue 101a) will have an entry in the active circuit memory 160a. The memory 160a is divided in sections according to message priority. For example, if there are two priority levels, "higher" and "lower", each active circuit memory would have two sections 160(H) and 160(L).

Under the control of a scheduler 135a, which is part of a controller 136a that regulates loading into and retrieval from the buffer memory, each of the interface units 54 cycles through the higher priorities entries in its active circuit memory 160a and, one-by-one, it performs the operations specified by the transmit and receive work queues corresponding to those entries. Thus, when interface unit 54a retrieves an entry from its transmit work queue 100a, it enters into the context memory the beginning virtual address in the main memory of the information to be transferred, and the number of bytes to be transferred. It then returns from the mapping memory 95a the physical addresses of beginning locations in the main memory 44a. The unit 54a then retrieves from the host main memory 44a the data in those locations and loads it into a group of segments (clusters of locations) in the buffer memory 120a. It then updates the context memory entry for that circuit. Specifically, it updates main memory 44a virtual address to the next address from which data is to be retrieved, and it decrements the amount of data to be transferred, by the size of the block retrieved from the memory 44a.

At a subsequent time, the scheduler 135a initiates retrieval of a cell from the block of control and/or data that is now resident in the buffer memory 120a. A packaging unit 138a having cell and framing capabilities constructs an ATM cell of the retrieved data. That is, an ATM header is appended to the data using well-known techniques. The packaging unit also performs all necessary network communication functions including execution of check sum routines to ensure data accuracy. At the frame level, the context memory 140a retains state information, such as the ATM

AAL-5, CRC and the frame protocol, and appends this information to the frame trailer per the ATM standard. A frame may or may not be extended over multiple cells that may or may not be intermixed with cells from different circuits.

At the receiving end, a parser 139b in the unit 54b interprets the header on the cell and causes the data portion of the cell to be loaded into the buffer memory 120b. Incoming ATM cells are entered one-by-one into the buffer 120b. Subsequently, the data is retrieved and transferred to specific physical memory locations in the host main memory 44b, as identified in the mapping memory 95b. When all transmit work queue descriptors for a given virtual circuit are completed and if no more incoming or outgoing data is present in the buffer memory 120a, then the corresponding entry in the active circuit memory 160b is removed. Clearly, the foregoing arrangement is efficient because, with the use of direct physical memory locations in main memories 44a and 44b, the transfer of control and data is made without the intervention of either of the host operating systems.

In a preferred buffer arrangement message data is transferred into and out of each network interface units 54a, 54b and 54c via the respective dynamic buffers 120a, 120b and 120c. The size of the buffer depends on the requirements and speed of the computer network. Typically, a RAM buffer having a size of 160 bits X 4 Kbits, i.e., a capacity of 640 Kbits will be sufficient. For management purposes, the memory is divided into segments, each of which contains a number, e.g., 20, of contiguous memory locations. In this example, each location contains a single byte of data.

The buffer arrangement includes, in addition to the buffer memory 120a, a thread memory 130a, a free list memory 150a and a context memory 140a. The thread memory contains, for each transmit circuit link and each receive circuit link occupying space in the buffer memory 120a, a thread, i.e., linked list, of the buffer segments occupied by the circuit, in the order in which data was entered into these segments. The earliest entry in the thread is at the "head" and the latest is at the "tail". The free list memory is a FIFO memory containing a list of available segments in the memory 120a. However *multiple* threads can exist simultaneously in a single memory, each having its own discrete series of pointers and links held in the context and thread

memories, respectively. If there is no data for a specific circuit in the buffer memory 120a, then no buffer memory or thread memory resources are allocated to that circuit and the head and tail pointer in the context memory 140a are nulled. This arrangement, thus, allocates resources with greater efficiency and enables, in general, a lower
5 total memory size.

When data is to be loaded into the memory 120a, the buffer control 136a retrieves free buffer memory segments, as needed, from the free list memory 150a, and loads the data into those buffer segments. It also updates the free list memory and the thread memory. Specifically, as each segment is filled, that segment's associated
10 thread memory is updated to point to the next available segment being allocated from the free list. It also updates the head address and byte count in the context memory 140a entry for that circuit.

When data is retrieved from the buffer memory 120a, the reverse procedure is followed. That is retrieval begins at the tail address of the associated thread. When
15 the retrieval operation is complete, the tail and byte count in the context memory 140a are updated. As segments are freed, they are added back to the free list in memory 150a and the associated thread memory is cleared.

In this embodiment, the context memory 140a is provided with the number of bytes in the dynamic buffer 120a that are utilized by each circuit link. The operation
20 of the thread memory 130a is monitored to derive the number of bytes for each virtual connection.

The interface system according to this invention makes possible the dynamic optimization of data transfers. That is, the system operates efficiently for large bulk transfers of data and for smaller transfers. This optimization is based in-part on the
25 establishment of priority and flow control standards.

Each writing operation into the dynamic buffer 120a is generally followed by a reading operation in which information is retrieved from the buffer. Higher priority retrievals (reads) are recognized first, followed by the lower priority retrievals. Within priority levels, the system cycles continuously through the active circuit en-
30 tries in "round-robin" fashion. In transmission of data, one ATM cell is transferred at

-11-

a time from the buffer memory to the network link 84a. In receiving data, variable blocks of data (typically 256 bytes) are transferred to the host, optimizing latency. When the buffer begins to saturate, then the data may be transferred to the host computer in larger blocks, , optimizing the transfer efficiency, to prevent memory over-
5 load. Ordinarily, this occurrence results from host saturation, so that access to the system bus is temporarily limited. Clearly the foregoing arrangement copes with the problem of overflow by using a more-efficient mode of transfer to the host main memory, i.e. larger block size.

10 The network interface units 54 operate as bus masters as the system busses of the respective host computers. In other words, they are capable of initiating operations on these buses. It should be noted, also, that each of the memories or registers 95a, 100a, 110a, 130a, 140a, 150a and 160a used in the network interface unit 54a is preferably a discrete physical memory so that a plurality of these memories can be
15 accessed simultaneously. This enhances the operating speed of the network interface unit 54a.

The above description has focused upon a data transfer between two discrete computers 30a and 30b. We shall now discuss the routing of data between a cluster of three or more linked computers. The processes described above remain unchanged.

20 A switch 82 (Fig. 1) is used to route communications among three or more computers. A preferred embodiment of the switch 82 is depicted in Fig. 4. The switch receives incoming signals through a set of serial-parallel converters 200 from a plurality of computers. Likewise, outgoing signals are routed to the computers through a series of outgoing parallel-serial converters 202. The incoming and outgoing signals can, of course, derive from and be routed to a variety of data processing
25 devices besides computers. For example, the switch 82 can be interconnected with one or more other switches. Furthermore, these incoming and outgoing functions can be provided by a single converter circuit such as the circuit 90 detailed in Fig. 2. The incoming signals are routed to a multiplexer (MUX) 204 that enables selection of a
30 single input to a dynamic buffer 206. Likewise, outgoing signals precede to a demul-

tiplexer (DEMUX) 208 that selects the appropriate outgoing converter 202. A MUX/DEMUX control circuit 212 governs routing of signals to and from the buffer based upon the operation of the dynamic buffer control circuit 210. Operation of the dynamic buffer is substantially similar to that of the network interface unit 54a. A
5 dynamic buffer control circuit 210 comprises a series of memories like those described with reference to the interface units of Figs 2 and 3.

Specifically, the switch includes a mapping memory that maps each incoming VCI to the outgoing VCI of the same circuit and to the output port for each outgoing VCI. The thread memory 214, the context memory 218 and the active circuit memory
10 are arranged in terms of either the incoming VCI's or the outgoing VCI's. The mapping memory 213 is interconnected with the context memory. When the switch controller establishes each circuit, it also instructs the switch context memory 218 and mapping memory 213 to establish a circuit indexed by the corresponding VCI. The context/mapping memory 218, 213 includes entries with respect to each VCI of the
15 head address, the tail address, the byte count, the flow control status and the transmission priority. Assuming that determination of the outgoing VCI takes place when the incoming cell is to be loaded into the buffer 206. The incoming VCI is replaced with the outgoing VCI that is mapped from the memory 213 before the cell is written into the dynamic buffer 206. Unlike the data processing device interfaces 54a, b and
20 c, in which physical addresses are mapped in connection with a given VCI, the switch 82 undertakes a direct VCI-to-VCI mapping.

While not shown, the switch 82 and each of the interface units include flow control circuitry that communicates with various computers and switches in the system to prevent excessive data inflow. These circuits can be implemented using well-
25 known arrangements and can be linked using a discrete flow control data line.

The foregoing has been a detailed description of a preferred embodiment of the invention. Various modifications and additions can be made without departing from the spirit and scope of the invention. For example, the buffer control system described can be varied and substituted for a variety of other acceptable buffer control
30 protocols. Such protocols should be able to transfer data based upon the establishment of virtual circuits and their corresponding identifiers according to this invention.

ATM is only one of a plurality of possible communication standards that can be employed . Other standards that are contemplated include the TCP standard. Likewise, other types of network switch can be used in conjunction with the network interface units of this invention. Finally, while the term "network interface unit" is used herein,
5 it is expressly contemplated that any acceptable platform can be utilized for the interface according to this invention and the interface can be provided as a permanent system in an overall computer, disk, peripheral or other data processing device. Accordingly, this description is means to be taken only by way of example and not to otherwise limit the scope of the invention.

CLAIMS

- 1 1. A computer interface for accomplishing a transfer from a transmitting application
2 running on a first computer, having a system bus and a first main memory connected
3 to said system bus, to a receiving application running on a second computer, said
4 transmitting application providing an operating system with transmit virtual addresses
5 in said main memory of data to be transmitted to said receiving application, said inter-
6 face comprising:
 - 7 A. a mapping memory;
 - 8 B. a transmit work queue memory;
 - 9 C. means in said operating system for:
 - 10 1. loading into said mapping memory the transmit physical ad-
11 dresses corresponding to said transmit virtual addresses, and
 - 12 2. entering into said transmit work queue memory a specification
13 of each data transmission, said specification including an identification of the transfer,
14 and an identification of a destination of said data, said identification of the transfer
15 also including identification of a location of each of said transmit virtual addresses in
16 said mapping memory;
 - 17 D. means for accessing said first main memory in accordance with said
18 physical addresses contained in said mapping memory to retrieve said data therefrom;
19 and
 - 20 E. means for transmitting said retrieved data to said second application.
- 1 2. The interface defined in claim 1 further including:
 - 2 A. a receive work queue memory;
 - 3 B. means in said operating system for interrogating a receiving applica-
4 tion running on said first computer for the identification of receive virtual addresses in

5 said first main memory into which received data transferred to that receiving applica-
6 tion from a transmitting application on said second computer is to be loaded,

7 C. means in said operating system for:

8 1. loading into said mapping memory receive physical addresses in said
9 first memory corresponding to said receive virtual addresses, and

10 2. entering into said receive work queue memory a specification of each
11 data transmission, said specification including an identification of the transfer, and an
12 identification of a destination of said data, said identification of the transfer also in-
13 cluding identification of a location of each of said receive virtual addresses in said
14 mapping memory,

15 D. means for receiving data, from said second computer, identified by said
16 transfer identification, and

17 E. means for accessing said first main memory by means of said receive
18 physical addresses contained in said mapping memory to load into those addresses
19 data received from said transmitting application on said second computer.

1 3. The interface defined in claim 1 wherein transmitting means includes a framing
2 and deframing unit that frames data for transmission to said second computer based
3 upon a predetermined communication standard.

1 4. The interface defined in claim 3 wherein the predetermined communication stan-
2 dard comprises the ATM standard.

1 5. The interface defined in claim 2 wherein the transmitting means includes a fram-
2 ing and deframing unit that frames the data for transmission to said second computer
3 based upon an ATM communication standard.

- 1 6. The interface defined in claim 5 wherein the identification of each data transfer
2 defines a virtual circuit identified by a respective virtual channel identifier, and
3 wherein the transmit work queue memory includes locations corresponding to each
4 virtual circuit.
- 1 7. The interface defined in claim 6 wherein the receive work queue memory includes
2 an entry corresponding to each virtual circuit.
- 1 8. The interface defined in claim 7 wherein the mapping memory includes a location
2 corresponding to each virtual circuit for storing respective of the transmit physical ad-
3 dresses.
- 1 9. The interface defined in claim 2 wherein each of the mapping memory, the
2 transmit work queue memory and the receive work queue memory is in the address
3 space of the system bus.
- 1 10. The interface defined in claim 6 further comprising a buffer memory for selec-
2 tively storing data received from the system bus and retrieving stored data for trans-
3 mission to the system bus, the buffer memory including a buffer controller that oper-
4 ates the buffer memory to store and retrieve data for each virtual circuit on a time-
5 shared basis.
- 1 11. The interface defined in claim 10 wherein the buffer memory is constructed and
2 arranged to selectively store data received from and retrieve data for transmission to
3 the second computer over a communication link and wherein the buffer controller is
4 constructed and arranged to cause the buffer memory to store data received from the
5 second computer for each virtual circuit on a time-shared basis.

1 12. The interface defined in claim 11 further comprising a thread memory containing
2 a thread defining a plurality of entries between a head entry and a tail entry, each of
3 the entries comprising an address representative of an active memory segment in the
4 buffer memory corresponding to each virtual circuit and the entries defining a linked
5 list of segment addresses, and further comprising means, in the buffer controller, for
6 adding entries to the thread memory and deleting entries from the thread memory
7 based, respectively, upon a loading into and emptying corresponding memory seg-
8 ments in the buffer memory.

1 13. The interface defined in claim 12 further comprising a free list memory having a
2 plurality of entries each corresponding to an unoccupied segment in the buffer mem-
3 ory that is free of data stored, and further comprising means in the buffer controller
4 for removing an entry having a predetermined segment address from the free list
5 memory, and concurrently establishing an entry having the predetermined segment
6 address in the thread memory.

1 14. The interface defined in claim 13 further comprising a context memory intercon-
2 nected with the buffer memory and the system bus, for storing a plurality of entries,
3 each of the entries corresponding to a predetermined virtual circuit, each of the entries
4 including, for the respective virtual circuit, information defining a data transfer prior-
5 ity level, the head location and the tail location of a thread of segment addresses in the
6 buffer memory and an amount of data contained in the buffer memory.

1 15. The interface defined in claim 2 further comprising an active circuit memory,
2 interconnected with the system bus, that stores entries identifying each data transfer
3 for which a specification is contained in at least one of the transmit work queue mem-
4 ory and the receive work queue memory, the active circuit memory including means
5 for continuously storing new entries as new data transfers are initiated and deleting
6 existing entries as existing data transfers are completed.

3 for which a specification is contained in at least one of the transmit work queue mem-
4 ory and the receive work queue memory, the active circuit memory including means
5 for continuously storing new entries as new data transfers are initiated and deleting
6 existing entries as existing data transfers are completed.

1 16. The interface defined in claim 15 wherein each of the entries in the active circuit
2 memory includes an identification of a respective priority level of the data transfer.

1 17. The interface defined in claim 2 wherein each of the transmit work queue mem-
2 ory and the receive work queue memory includes a respective pointer register, each
3 pointer register having an entry corresponding to each specification in the respective
4 transmit work queue memory and receive work queue memory each entry including a
5 pointer representative of a location in the respective transmit work queue memory and
6 receive work queue memory of the specification for the next data transfer to occur.

1 18. A switch for accomplishing transfers of data from transmitting applications run-
2 ning on transmitting computers to receiving applications running on receiving com-
3 puters, each transmitting computer including means for framing the data with a virtual
4 circuit identifier, said switch comprising:

5 A. a plurality of input ports for reception of data from transmitting com-
6 puters;

7 B. a plurality of output ports for transmission of data to receiving comput-
8 ers;

9 C. a mapping memory containing entries relating virtual circuit identifi-
10 ers to output ports;

11 D. a switch buffer memory for selectively storing data received from the
12 transmitting computer and for retrieving stored data for transmission to a receiving

13 computer, the switch buffer memory including a switch buffer controller that operates
14 the memory to store and retrieve data based upon the virtual circuit identifiers;

15 E. a switch thread memory containing a thread defining a plurality of en-
16 tries between a head entry and a tail entry, each of the entries comprising an address
17 representative of an active memory segment in the switch buffer memory correspond-
18 ing to each virtual circuit and the entries defining a linked list of segment addresses,
19 and further comprising means, in the switch buffer controller, for adding entries to the
20 thread memory and deleting entries from the thread memory based, respectively, upon
21 a loading into and emptying corresponding memory segments in the switch buffer
22 memory;

23 F. a switch free list memory having a plurality of entries each correspond-
24 ing to an unoccupied segment in the switch buffer memory that is free of data stored,
25 and further comprising means in the switch buffer controller for removing an entry
26 having a predetermined segment address from the free list memory, and concurrently
27 establishing an entry having the predetermined segment address in the thread mem-
28 ory;

29 G. a context memory, interconnected with the switch buffer memory, for
30 storing a plurality of entries, each of the entries corresponding to a predetermined
31 virtual circuit identifier, each of the entries including, for the respective virtual circuit
32 identifier, information defining a data transfer priority level, the head location and the
33 tail location of a thread of segment addresses in the switch buffer memory and an
34 amount of data contained in the switch buffer memory; and

35 H. means for directing data retrieved from said buffer memory to the output
36 port relating to the virtual circuit identifier associated with the data.

1 19. The switch defined in claim 18 further comprising a switch active circuit mem-
2 ory, interconnected with the switch buffer controller, that stores entries identifying
3 each virtual circuit for which data is contained in the switch buffer memory, the
4 switch active circuit memory including means for continuously storing new entries as

-20-

5 new data transfers are initiated and deleting existing entries as existing data transfers
6 are completed.

1 20. The switch defined in claim 20 wherein each of the entries in the switch active
2 circuit memory includes an identification of a respective priority level of the data
3 transfer.

1 21. A method for accomplishing a transfer from a transmitting application running on
2 a first computer, having an internal bus and a first main memory connected to said
3 system bus, to a receiving application running on a second computer, said transmitting
4 application providing an operating system with transmit virtual addresses in said main
5 memory of data to be transmitted to said receiving application, said interface compris-
6 ing:

7 A. providing a mapping memory;

8 B. providing a transmit work queue memory;

9 C. by means of software running on said first computer:

10 1. loading into said mapping memory the transmit physical ad-
11 dresses corresponding to said transmit virtual addresses, and

12 2. entering into said transmit work queue memory a specification
13 of each data transmission, said specification including an identification of the transfer,
14 and an identification of a destination of said data, said identification of the transfer
15 also including identification of a location of each of said transmit virtual addresses in
16 said mapping memory;

17 D. accessing said first main memory in accordance with said physical ad-
18 dresses contained in said mapping memory to retrieve said data therefrom; and

19 E. transmitting said retrieved data to said second application.

1 22. The method defined in claim 21 further comprising:

2 A. providing a receive work queue memory,

3 B. interrogating, by means of software running on said first computer, a
4 receiving application running on said first computer for the identification of receive
5 virtual addresses in said first main memory into which received data transferred to
6 that receiving application from a transmitting application on said second computer is
7 to be loaded,

8 C. by means of said software system:

9 1. loading into said mapping memory receive physical addresses in said
10 first memory corresponding to said receive virtual addresses, and

11 2. entering into said receive work queue memory a specification each data
12 transmission, said specification including an identification of the transfer, and an
13 identification of a destination of said data, said identification of the transfer also in-
14 cluding identification of a location of each of said receive virtual addresses in said
15 mapping memory,

16 D. receiving data, from said second computer, identified by said transfer
17 identification, and

18 E. accessing said first main memory by means of said receive physical
19 addresses contained in said mapping memory to load into those addresses data re-
20 ceived from said transmitting application on said second computer.

1 23. The method defined in claim 22 wherein each of the step of transmitting and the
2 step of accessing includes framing the data according to an ATM communication
3 standard, including establishing a virtual channel identifier for each data transfer.

- 1 24. The method as defined in claim 24 further comprising storing into and retrieving
- 2 data from a buffer memory, the steps of storing and transferring being based upon the
- 3 virtual channel identifier corresponding to each data transfer, and further including
- 4 prioritizing the steps of storing and transferring based upon priority criteria associated
- 5 with each corresponding virtual channel identifier.

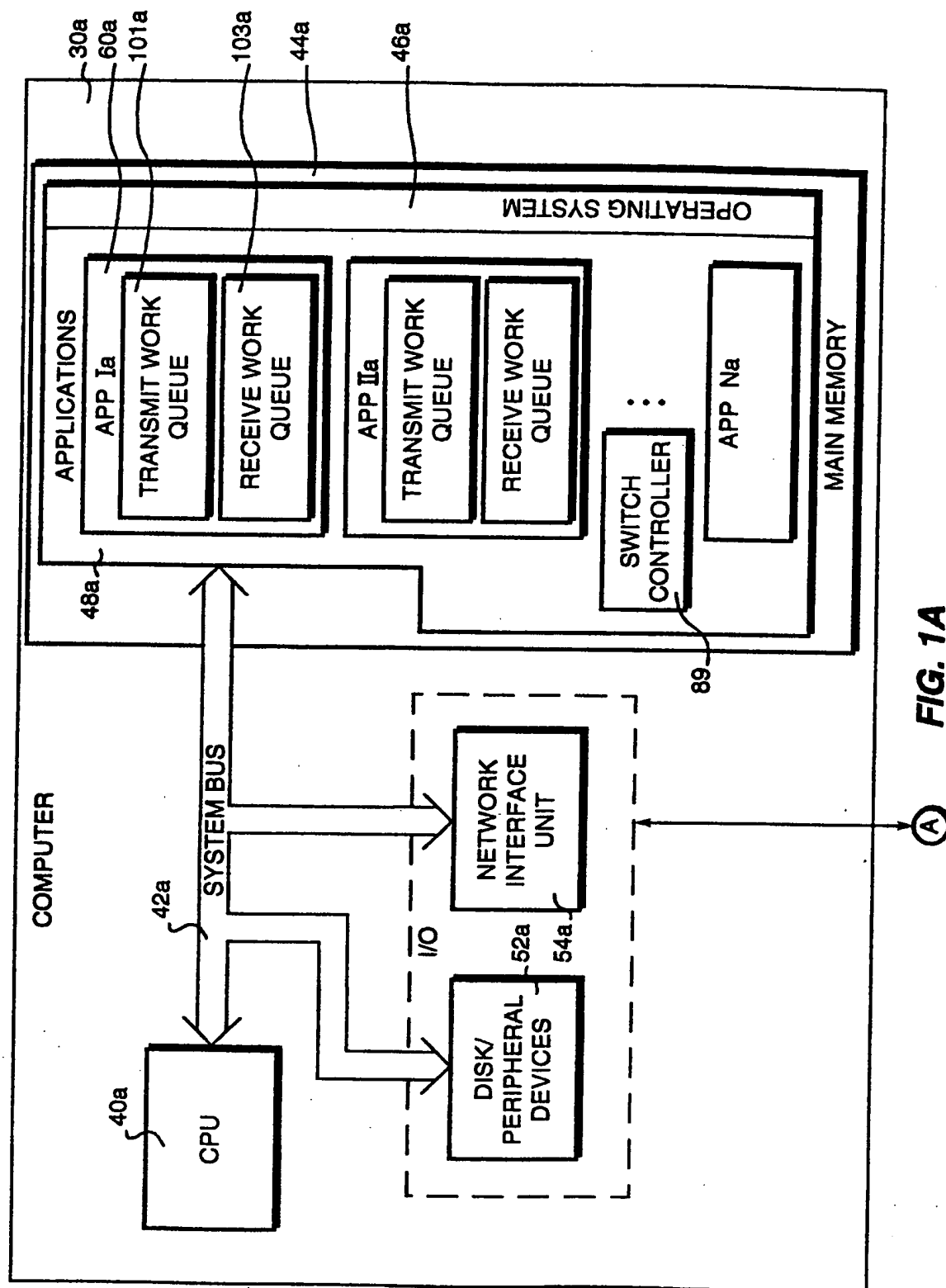
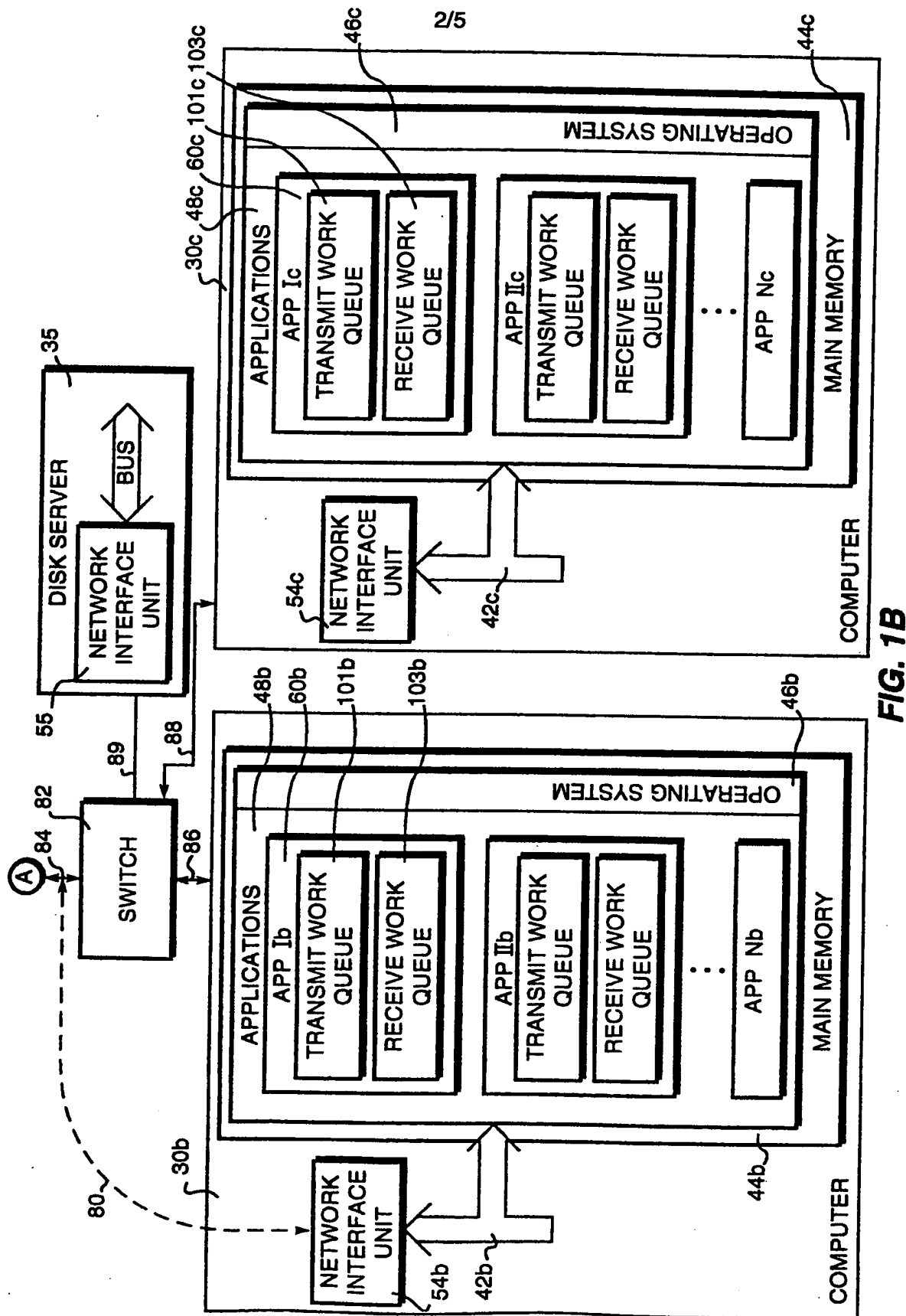


FIG. 1A



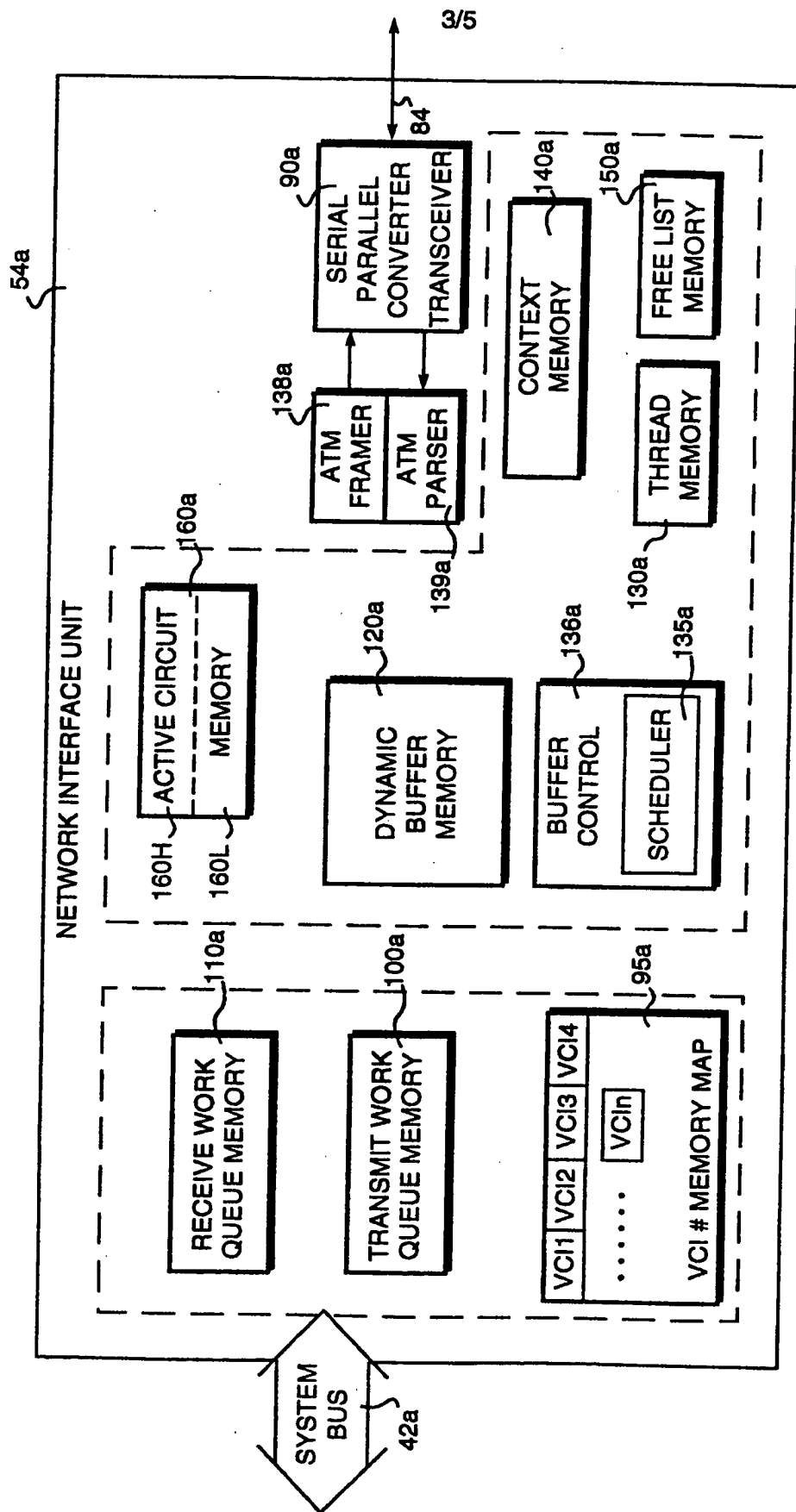


FIG. 2

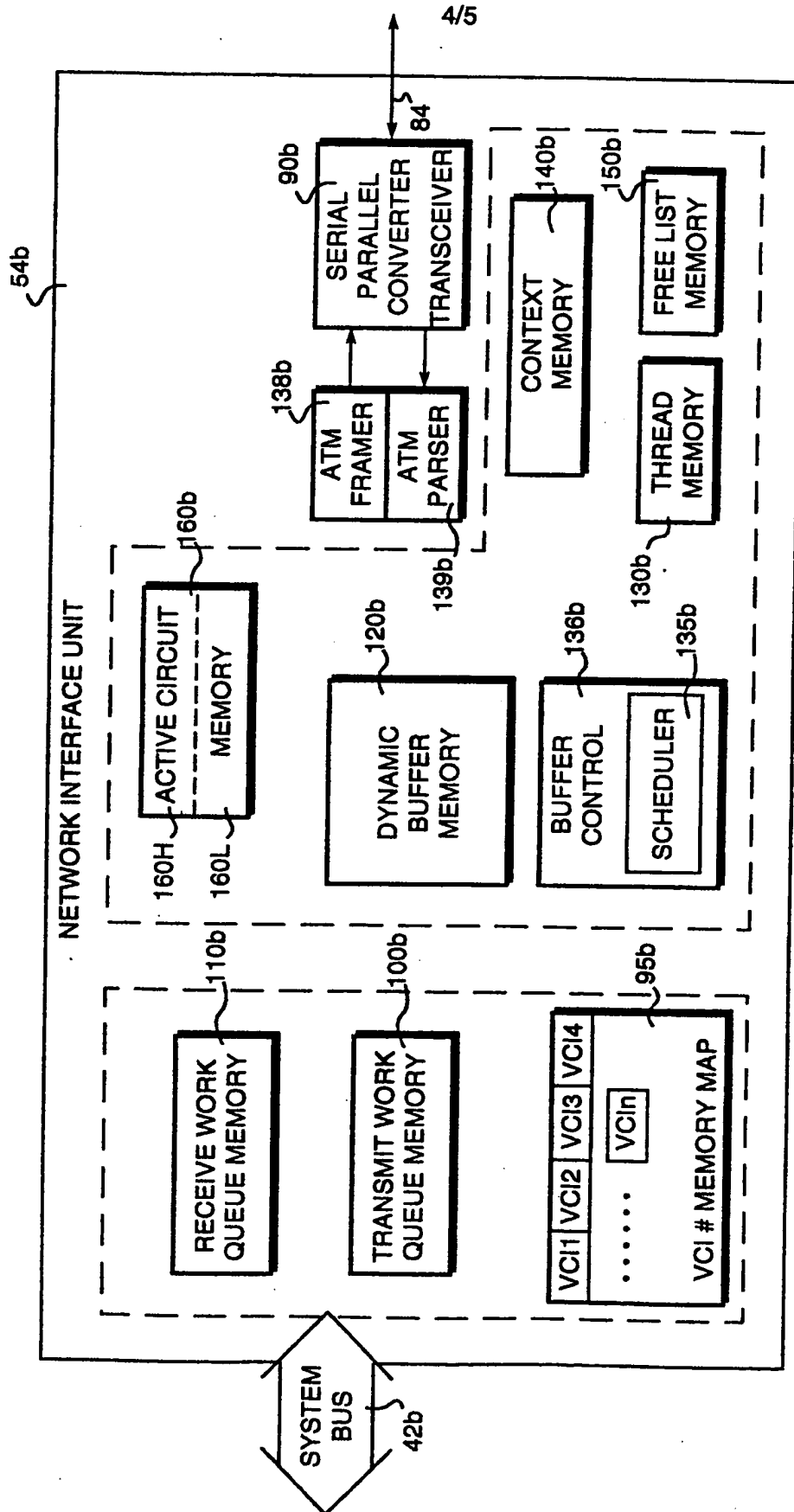


FIG. 3

5/5

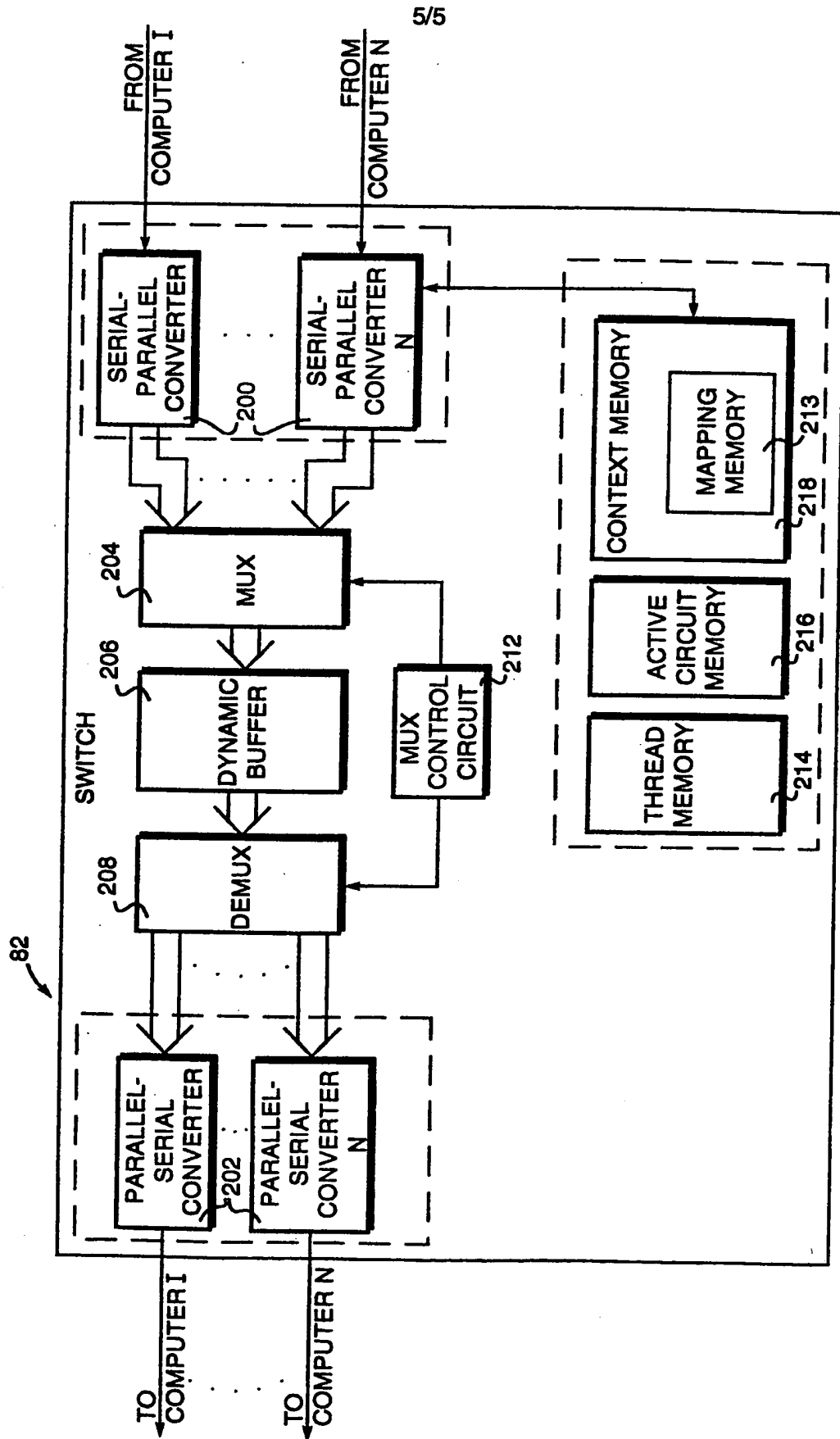


FIG. 4

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☒ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.